SIMULATION AND IMPLEMENTATION OF SEQUENTIAL LOGIC CIRCUITS

AIM:

To simulate and synthesis SR, JK, T, D - FLIPFLOP, COUNTER DESIGN using Xilinx ISE.

APPARATUS REQUIRED:

Xilinx 14.7 Spartan6 FPGA

PROCEDURE:

STEP:1

Start the Xilinx navigator, Select and Name the New project.

STEP:2

Select the device family, device, package and speed.

STEP:3

Select new source in the New Project and select Verilog Module as the Source type.

STEP:4

Type the File Name and Click Next and then finish button. Type the code and save it.

STEP:5

Select the Behavioral Simulation in the Source Window and click the check syntax.

STEP:6

Click the simulation to simulate the program and give the inputs and verify the outputs as per the truth table.

STEP:7

Select the Implementation in the Sources Window and select the required file in the Processes Window.

STEP:8

Select Check Syntax from the Synthesize XST Process. Double Click in the FloorplanArea/IO/Logic-Post Synthesis process in the User Constraints process group. UCF(User constraint File) is obtained.

STEP:9

In the Design Object List Window, enter the pin location for each pin in the Loc column Select save from the File menu.

STEP:10

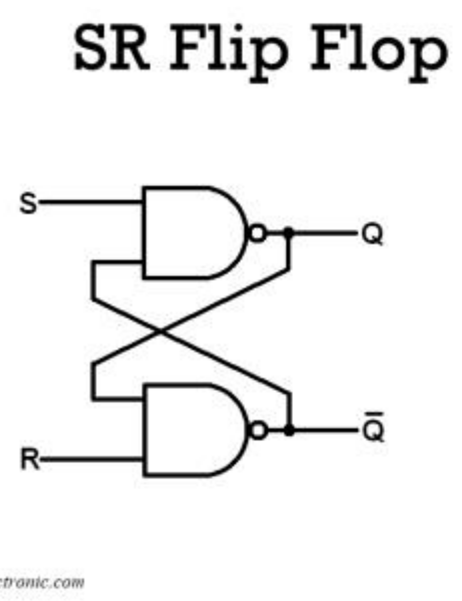
Double click on the Implement Design and double click on the Generate Programming File to create a bitstream of the design.(.v) file is converted into .bit file here.

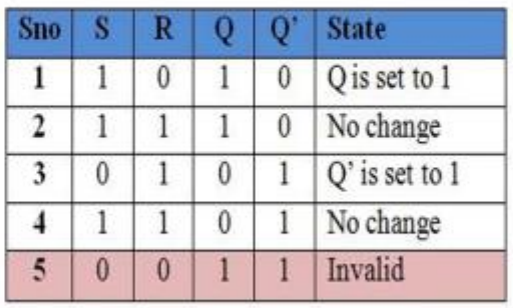
STEP:11

On the board, by giving required input, the LEDs starts to glow light, indicating the output.

LOGIC DIAGRAM:

SR FLIPFLOP:



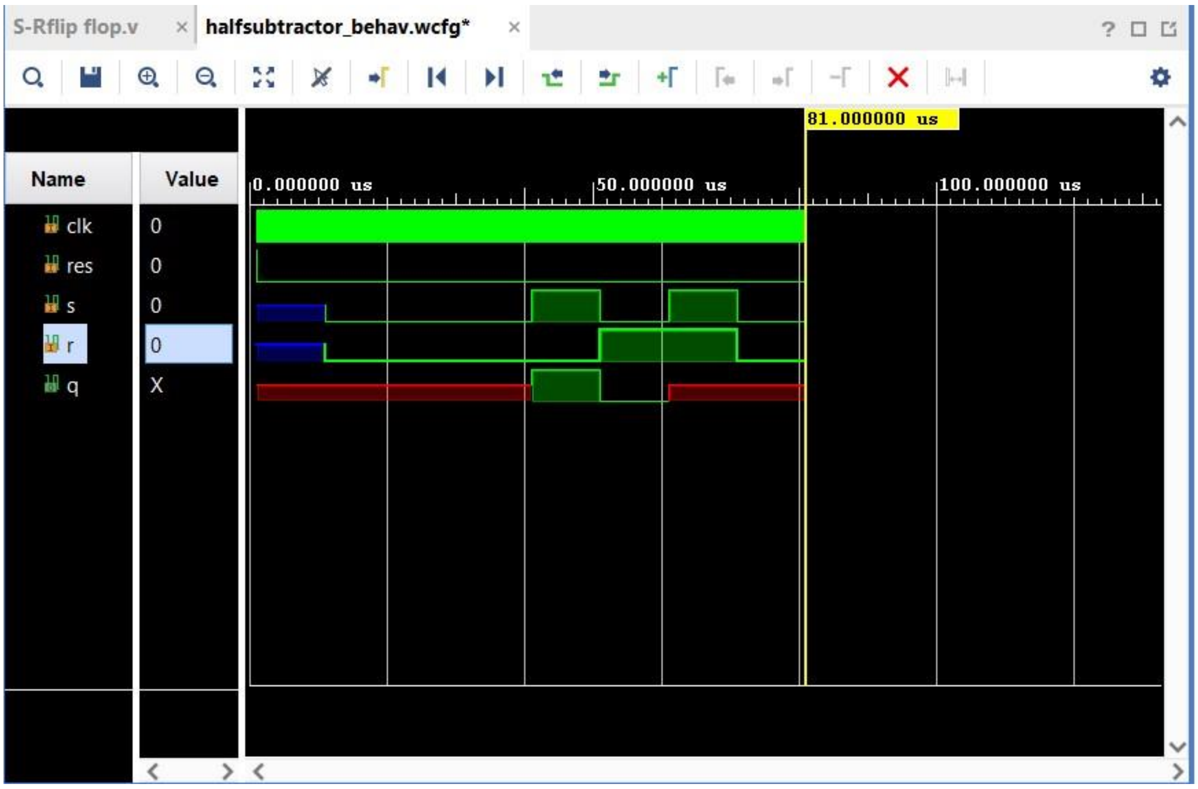


Sno	S	R	Q	Q'	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q' is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

VERILOG CODE:

```
 module srflipflop(clk,res,s,r,q);
input clk,res,s,r;
output reg q;
always@(posedge clk)
begin
if(res)
q<=1'b0;
else begin
case({s,r})
2'b00:q<=q;
2'b01:q<=1'b0;
2'b10:q<=1'b1;
2'b11:q<=1'bx;
default:q<=1'bx;
endcase
end
end
endmodule
```

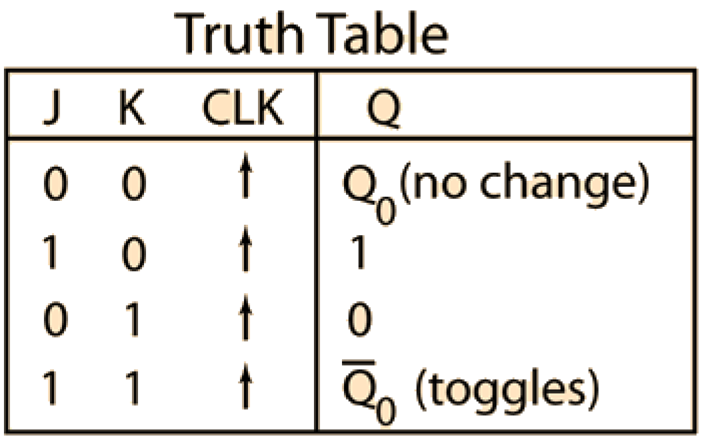
OUTPUT WAVEFORM :



Name	Value
clk	0
res	0
s	0
r	0
q	X

LOGIC DIAGRAM:

JK FLIPFLOP:



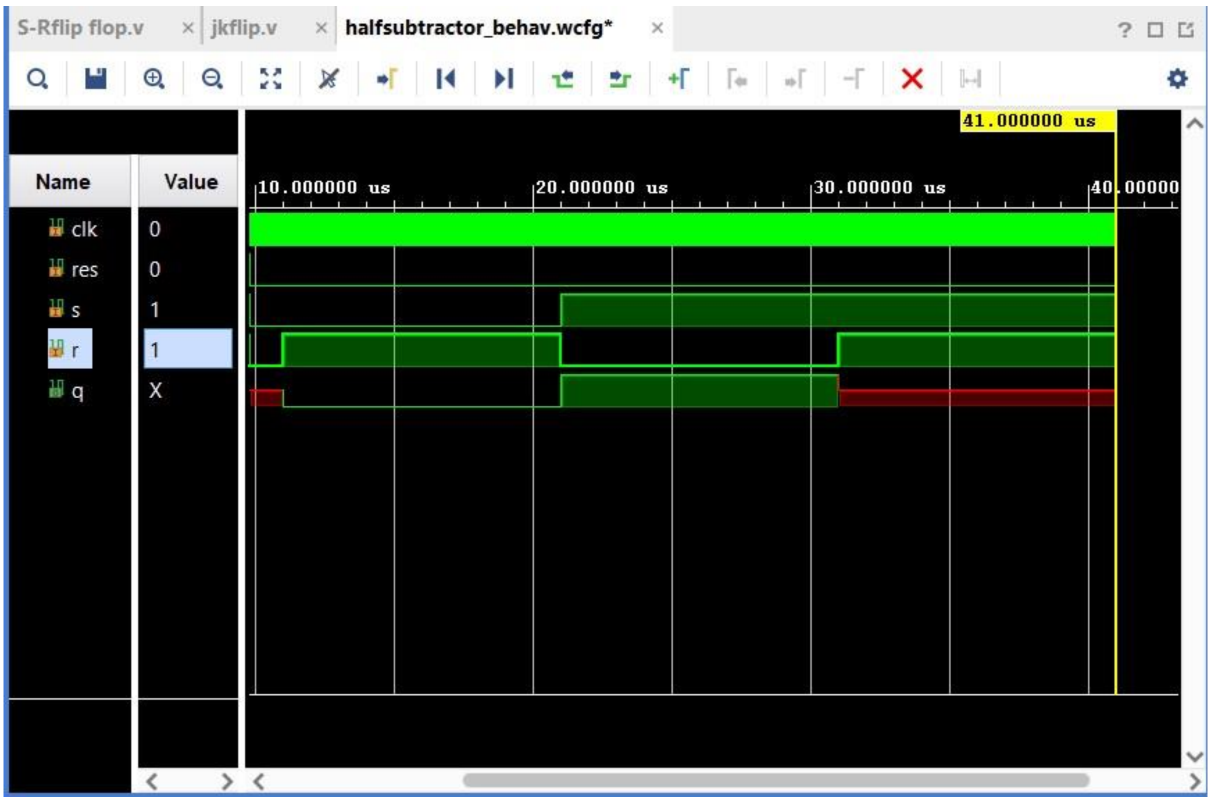
Truth Table

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q}_0$ (toggles)

VERILOG CODE:

```
 module jk(clk,res,j,k,q);
input clk,res,j,k;
output reg q;
always@(posedge clk)
begin
if(res)
q<=1'b0;
else begin
case({j,k})
2'b00:q<=q;
2'b01:q<=1'b0;
2'b10:q<=1'b1;
2'b11:q<=~q;
default:q<=~q;
endcase
end
end
endmodule
```

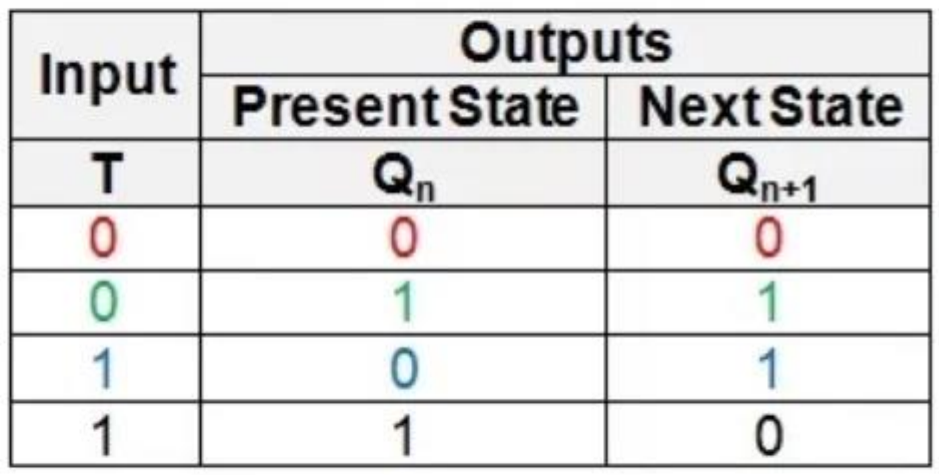
OUTPUT WAVEFORM :



Name	Value
clk	0
res	0
s	1
r	1
q	X

LOGIC DIAGRAM:

T FLIPFLOP:



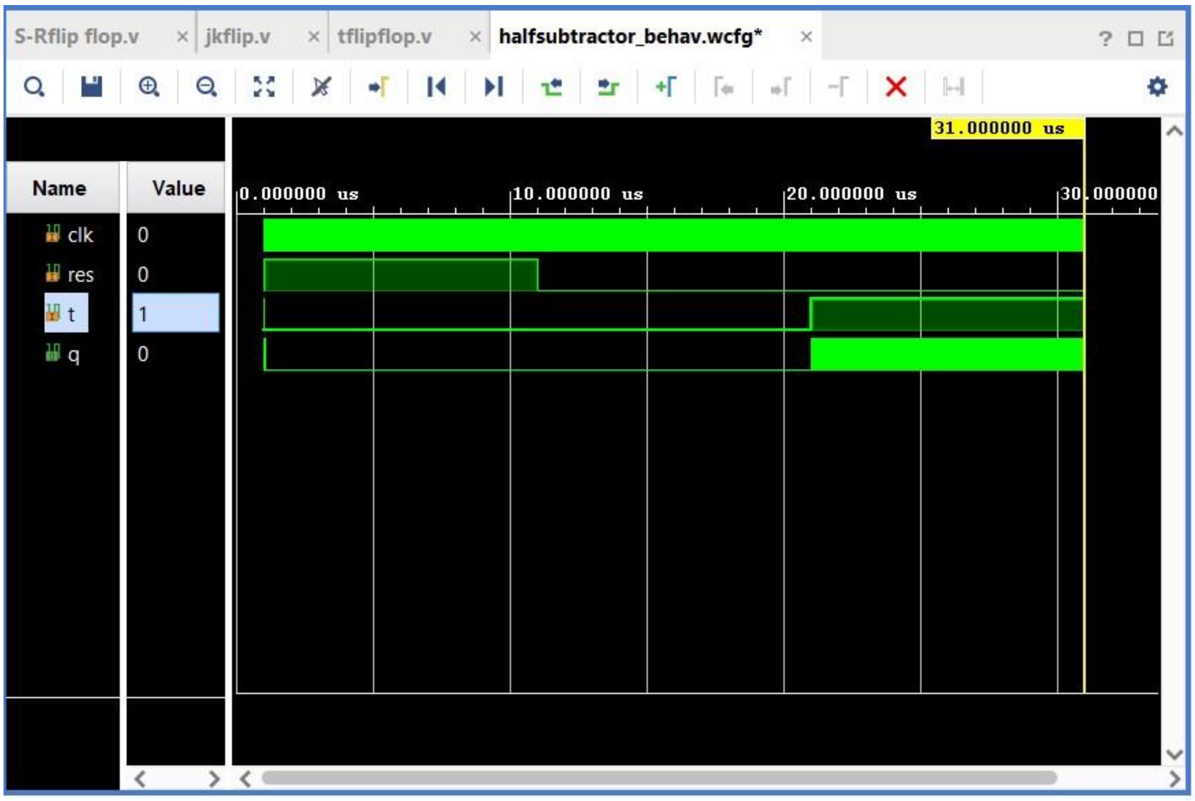
Input	Outputs	
	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

VERILOG CODE:

```
module tflipflop(clk,res,q,t);
input clk,res,t;
output reg q;
always@(posedge clk)
begin
if(res)
q<=1'b0;
else
begin
if(res)
q<=0;
else if(t)
q<=~q;
else
q<=q;
```

```
end
end
endmodule
```

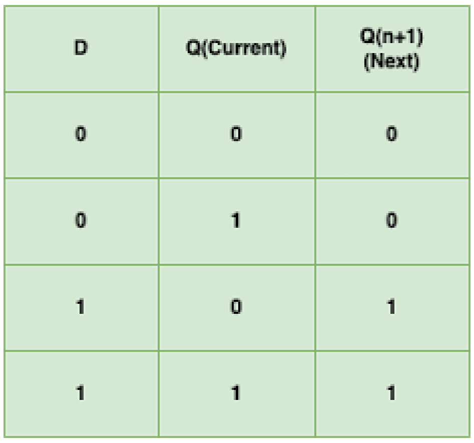
OUTPUT WAVEFORM :



Name	Value
clk	0
res	0
t	1
q	0

LOGIC DIAGRAM:

D FLIPFLOP:

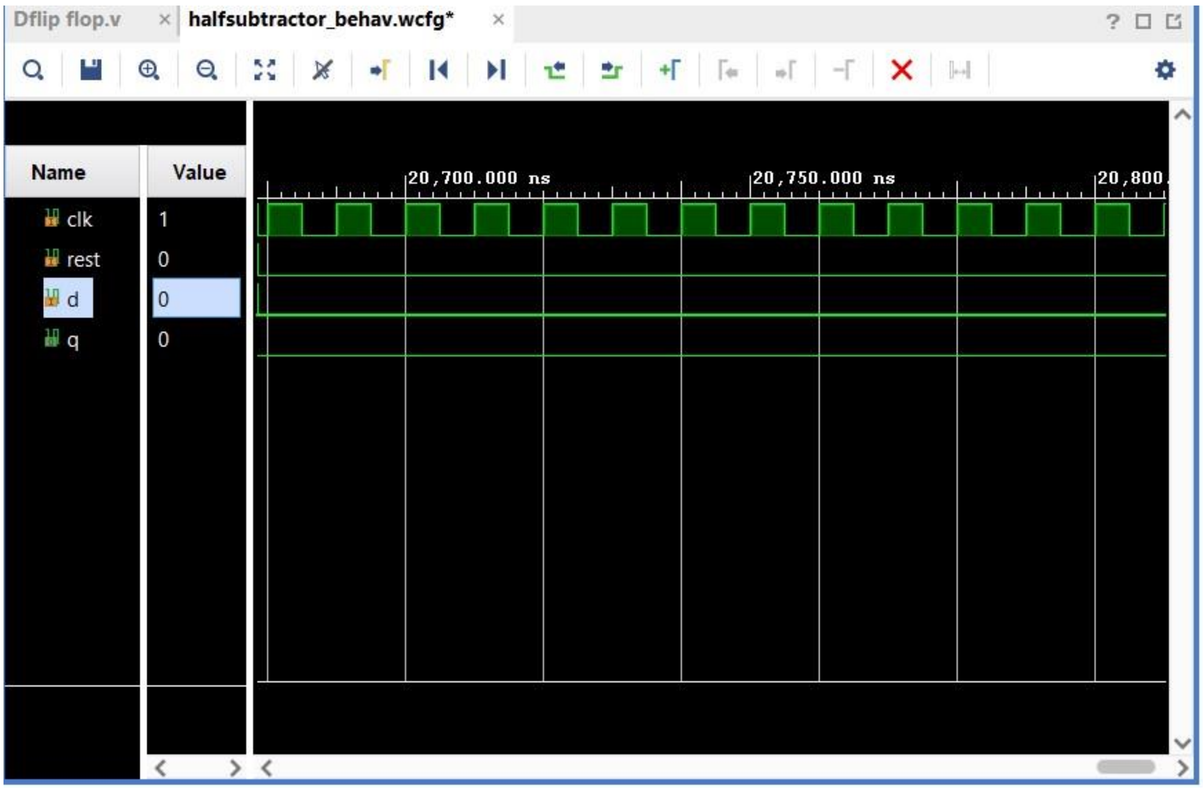


D	Q(Current)	Q(n+1) (Next)
0	0	0
0	1	0
1	0	1
1	1	1

VERILOG CODE:

```
module dflipflop(c,r,d,q);
input c,r,d;
output reg q;


always @(posedge c)
begin
if(r)
q<= 1'b0 ;
else
q<=d;
end
endmodule
```

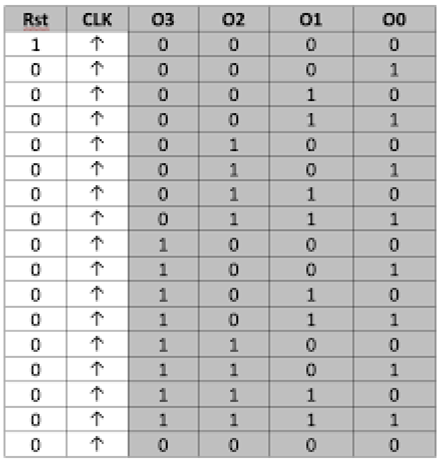
OUTPUT WAVEFORM :



Name	Value
clk	1
rest	0
d	0
q	0

LOGIC DIAGRAM:

COUNTER:



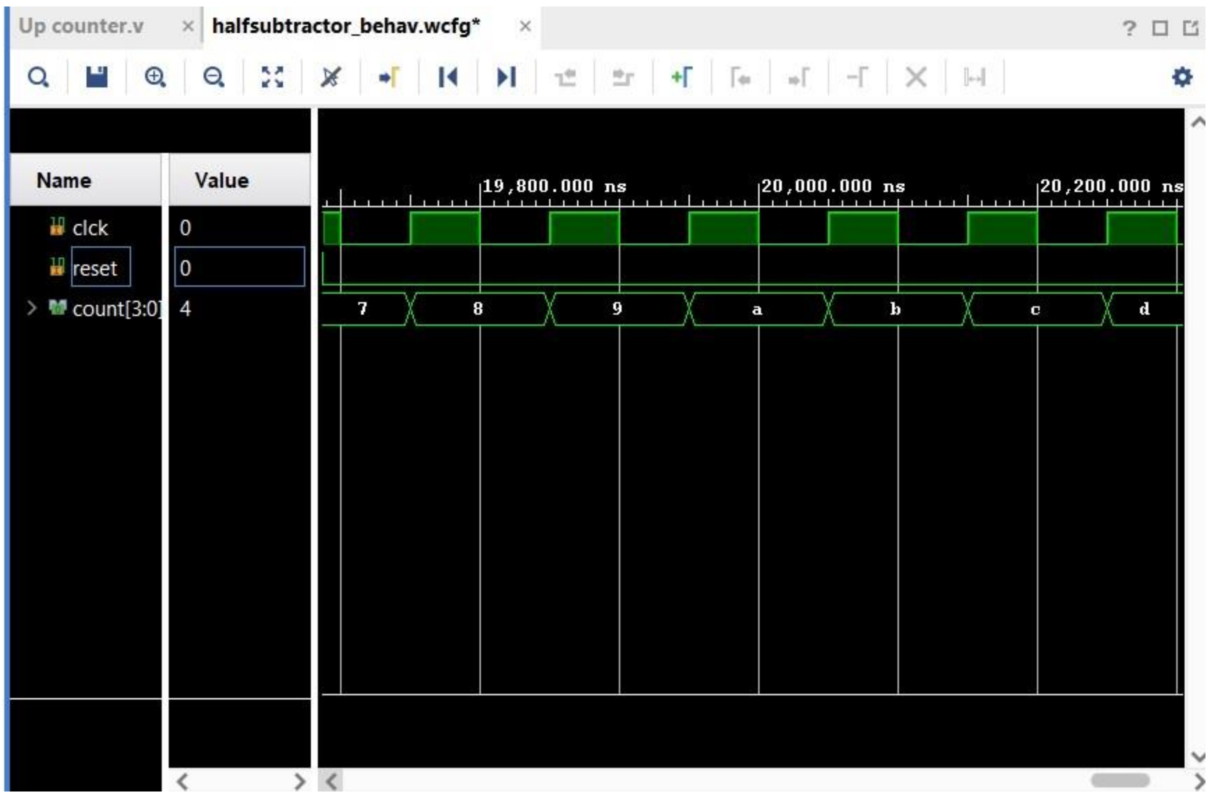
Rst	CLK	O3	O2	O1	O0
1	↑	0	0	0	0
0	↑	0	0	0	1
0	↑	0	0	1	0
0	↑	0	0	1	1
0	↑	0	1	0	0
0	↑	0	1	0	1
0	↑	0	1	1	0
0	↑	0	1	1	1
0	↑	1	0	0	0
0	↑	1	0	0	1
0	↑	1	0	1	0
0	↑	1	0	1	1
0	↑	1	1	0	0
0	↑	1	1	0	1
0	↑	1	1	1	0
0	↑	1	1	1	1
0	↑	0	0	0	0

VERILOG CODE:

UP COUNTER:

```
module upcounter(clk,rest,count);
input clk,rest;
output [3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rest)
count<=4'b0;
else
count<=count+1;
end
endmodule
```

OUTPUT WAVEFORM :



VERILOG CODE:

DOWN COUNTER:

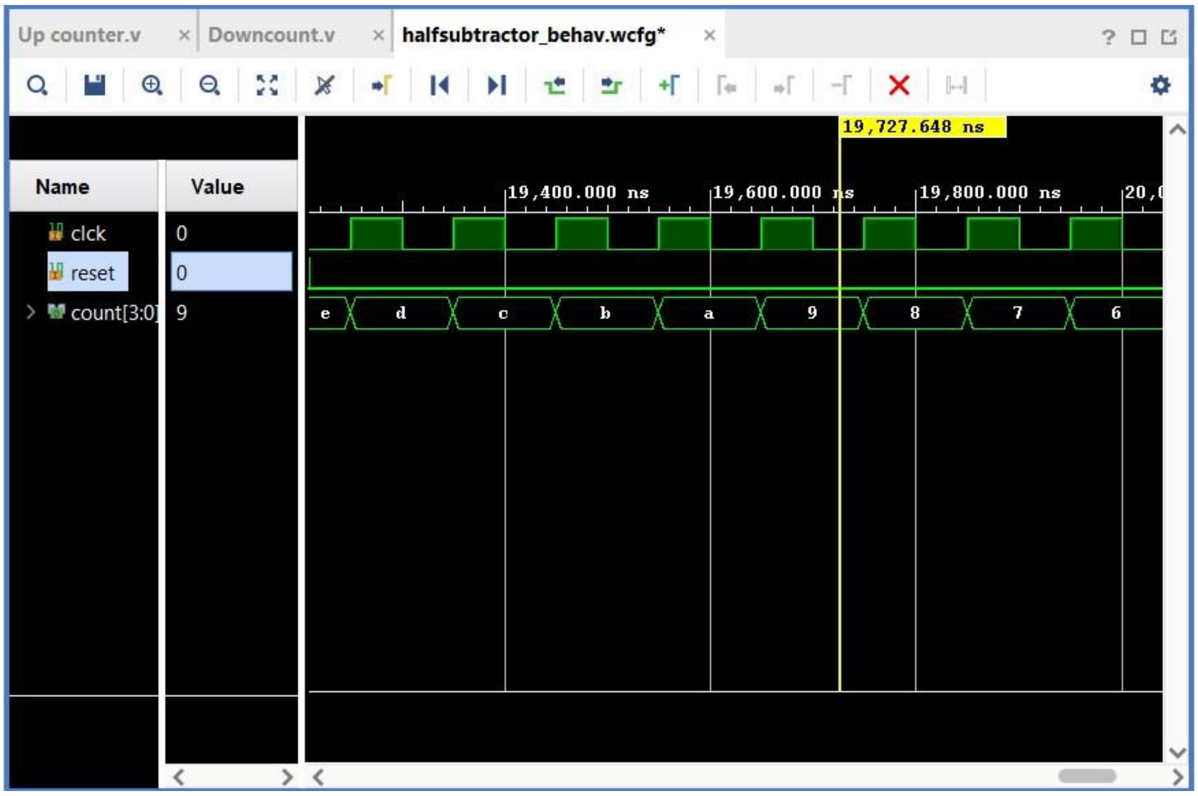
```
module downcounter(clk,rest,count);
input clk,rest;
output [3:0]count;
reg[3:0]count;
always@(posedge clk)
begin
if(rest)
count<=4'b0;
else
```

count<=count-1;

end

endmodule

OUTPUT WAVEFORM :



VERILOG CODE:

MOD 10:

module mod10(clk,res,count);

input clk,res;

output [3:0]count;

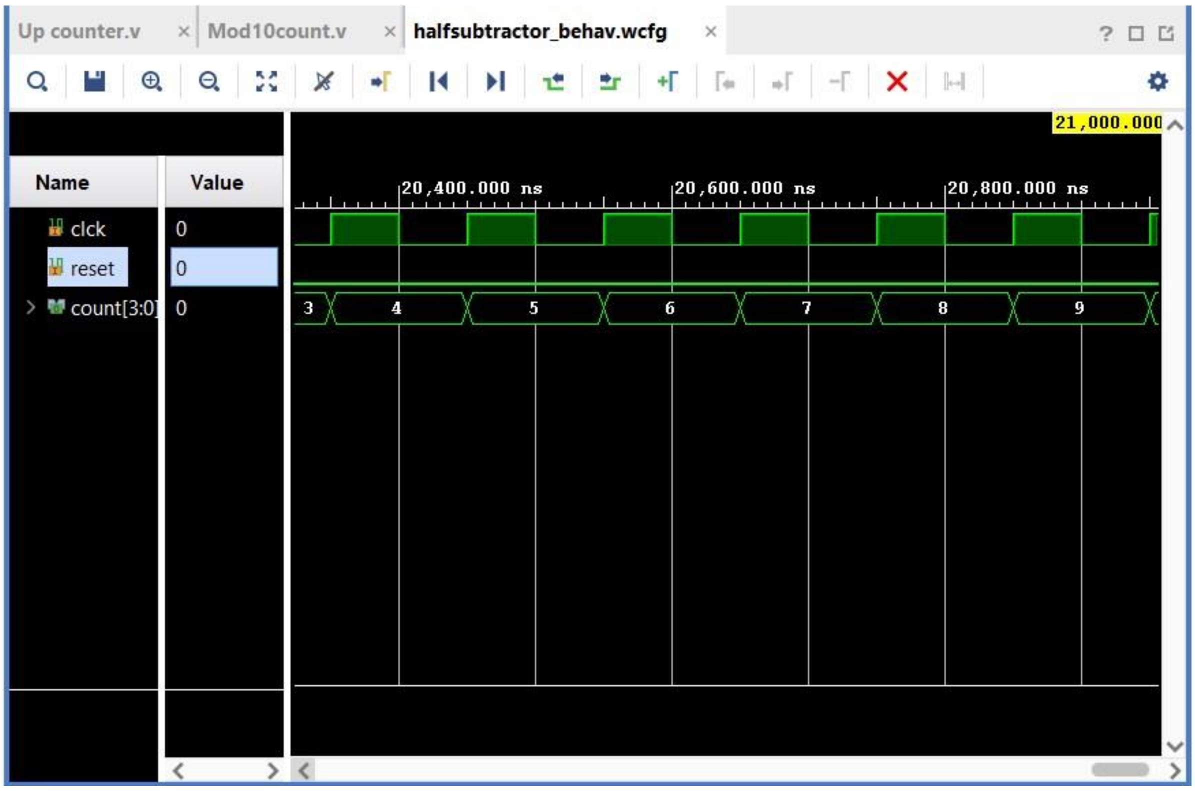
reg [3:0]count;

always@(posedge clk)

begin

```
if(res|count==4'd9)

count<=4'b0;

else

count<=count+1;

end

endmodule
```

OUTPUT WAVEFORM :



Name	Value
clck	0
reset	0
count[3:0]	0

VERILOG CODE:

RIPPLE CARRY COUNTER:

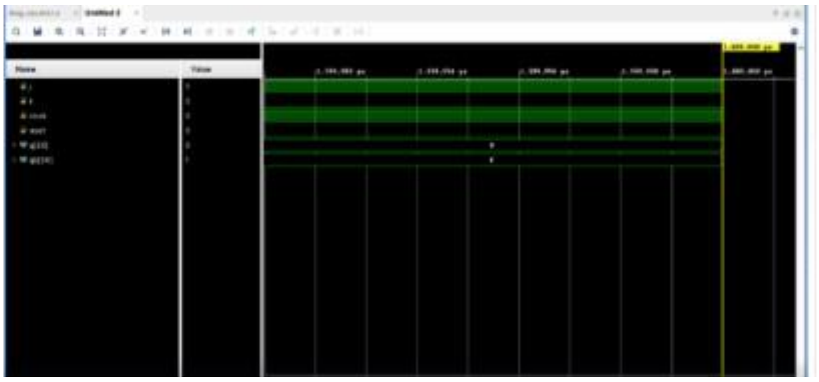
Verilog Code for jkff: (Behavioural model)

```
module jkff(j,k,clock,reset,q,qb);
input j,k,clock,reset;
output reg q,qb;
always@(negedge clock)
begin
case({reset,j,k})
3'b100 :q=q;
3'b101 :q=0;
3'b110 :q=1;
3'b111 :q=~q;
default :q=0;
endcase
qb<=~q;
end
endmodule
```

Verilog Code for 4-bit Asynchronous up counter using JK-FF (Structural model):

```
module ripple_count(j,k,clock,reset,q,qb);
input j,k,clock,reset;
output wire [3:0]q,qb;



jkff JK1(j,k,clock,reset,q[0],qb[0]);
jkff JK2(j,k,q[0],reset,q[1],qb[1]);
jkff JK3(j,k,q[1],reset,q[2],qb[2]);
jkff JK4(j,k,q[2],reset,q[3],qb[3]);
endmodule
```

OUTPUT WAVEFORM :



RESULT:

The simulation and synthesis SR FLIPFLOP, JK FLIPFLOP, T FLIPFLOP, D FLIPFLOP, COUNTERS by running successfully using Xilinx ISE.